

# CLASSIFICATION-ERROR COST MINIMIZATION STRATEGY: DCMS

*Devi Parikh and Tsuhan Chen*

Carnegie Mellon University  
Department of Electrical and Computer Engineering  
Pittsburgh, PA, USA  
{dparikh,tsuhan}@cmu.edu

## ABSTRACT

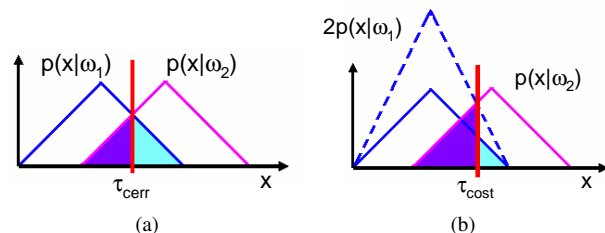
Several classification applications such as intrusion detection, biometric recognition, etc. have different costs associated with different classification errors. In such scenarios, the goal is to minimize the cost incurred, and not the classification error rate itself. This paper proposes a *Cost Minimization Strategy*, *dCMS*, which when applied to classifiers, provides a boost in the performance by reducing the cost incurred due to classification errors. *dCMS* is classifier-type independent, however it exploits the statistical properties of the trained classifier. It does not require classifiers to be retrained, which is particularly advantageous in scenarios where the costs vary dynamically. Convincing results are provided which indicate the statistically significant reduction in cost incurred by applying *dCMS*, in a diverse set of classification scenarios with datasets and classifiers of varying complexities.

**Index Terms**— cost minimization, *dCMS*, intrusion detection, volatile organic compounds, optical character recognition, Learn++, combining classifiers

## 1. INTRODUCTION

Several applications such as biometric recognition, intrusion detection, etc. require striking a balance in the trade-off between, for example, user convenience and integrity of the system. Due to this, different classification errors incur different costs. For example, in a high-security federal facility, the cost of permitting an imposter is much higher than the cost of denying access to someone genuine. On the other hand, consider the fingerprint recognizer on a consumer laptop which is at most used by a few people. In this case, there is a higher cost for the user's inconvenience due to false rejects. In such scenarios it is crucial to develop classification systems that minimize the cost incurred due to classification errors, and not the classification error rate itself.

Most often, unprincipled ad hoc techniques are used to deal with these costs. Slightly more principled techniques include using the Receiver Operating Characteristic (ROC) curves or precision-recall curves and choose a desired operating point. However, this works only for two class problems



**Fig. 1.** Depiction of Bayes rule based optimum strategy for classification for two class problem, with known underlying class conditional distributions with (a) same (b) different costs associated with both types of classification errors.

such as verification problems in biometric recognition [1] and there are no fully developed extensions of ROC curves to multiple classes [2]. Algorithms such as cost-sensitive Adaboost [3] have also been proposed, however if the costs involved vary, they require retraining the entire classification system.

Our proposed cost minimization strategy, *dCMS*, provides a principled approach that:

1. Can deal with multiple class problems
2. Is a simple post-training step, and hence does not require retraining of classifiers if the costs involved vary
3. Is classifier independent, and can be applied to any classification system that provides a confidence score in its classification

The rest of the paper is organized as follows. Section 2 describes the *dCMS* approach, Section 3 discusses the experiments conducted and results obtained and finally Section 4 concludes the paper.

## 2. APPROACH

To motivate our proposed approach, let us consider a classification problem with one feature,  $x$ , and two classes,  $\omega = \{\omega_1, \omega_2\}$ , where the conditional distributions of the two classes,

$p(x|\omega_1), p(x|\omega_2)$  are known, as depicted in Fig.1(a). An optimum operating point,  $\tau_{\text{cost}}$ , a threshold on the feature value such that the classification error rate, shaded regions in Fig.1(a), is minimized, can be determined using the Bayes classification rule [4] as depicted.

Suppose the cost of a false positive for  $\omega_2$  is twice that of  $\omega_1$ , then the optimum threshold  $\tau_{\text{cost}}$  that minimizes the cost incurred by the classification errors, shaded portion in Fig.1(b), can be determined using a similar Bayes classification rule for costs [4] as depicted in Fig.1(b). From here on, the optimum threshold will refer to the one which minimizes the cost incurred, and not the classification error rate itself.

The task is to now apply the above principle to a classification problem with different (known) costs associated with different classification errors with:

1. Unknown underlying class conditional distributions
2. High dimensional data
3. Multiple classes

The following subsections describe our approach in extending the above principle to each one of these three issues. While (1) and (2) have been well addressed by the community, the contribution of this work is to address (3).

### 2.1. Unknown distributions

In a one-feature problem, if the underlying class conditional distributions are not known, one can estimate them with a histogram. And using the Bayes classification rule [4] as depicted in Fig.1(b), an optimum threshold can be determined. It should be noted that since the distributions are now represented non-parametrically, the optimum threshold cannot be determined analytically. The search process, however, being just a one-dimensional search, is fairly simple. A set of possible threshold values (in the range of 0 to 1 for normalized outputs of classifiers) can be evaluated to determine the optimum threshold. To evaluate each threshold value, the cost of classification errors by using that threshold for classification is to be computed. Having computed the class conditional (cumulative) distributions via histograms and given the cost matrix indicating the cost of each type of classification error, computing this overall classification cost for every threshold is cheap.

### 2.2. High dimensional data

The above approach works for unknown class conditional distributions of one-dimensional data. However, with high dimensional data and unknown class conditional distributions, there is seldom enough data or resources to estimate these distributions (hence the need for learning algorithms!). Hence, a classification system is trained on this high dimensional data that provides a score that represents the classification

system's confidence that the instance belongs to each of the two classes. This score can now be perceived to be the one-dimensional feature on the basis of which a class is to be picked. The high-dimensional classification problem has now been converted to a one-dimensional problem for which the class conditional distributions can be estimated by histograms as described in Section 2.1, and the optimum threshold can be determined. Thus, we have resolved issues (1) and (2).

### 2.3. Multiple classes

So far we have considered a two-class problem. What if we have multiple (say  $C$ ) classes? We would still train a  $C$  class classification system that provides a confidence for an instance belonging to each of the  $C$  classes. This may be implemented as one classifier with  $C$  outputs or as  $C$  individual classifiers trained to respond to each of the  $C$  classes. From here on we will refer to the  $C$  outputs of the classification system, irrespective of the underlying classification system layout. If we can determine  $C$  optimum thresholds to be placed on each of the  $C$  outputs, the class corresponding to the output that exceeds its respective threshold can be picked as the classification decision. If multiple outputs exceed their thresholds, voting resolution techniques such as weighted majority voting, etc. [5] can be employed. Although we have solved this for  $C = 2$  in the previous subsections, applying the principle in Fig.1(b) given a  $C \times C$  cost matrix and  $C$  class conditional distributions is not straightforward. We resolve this by subdividing the problem into  $C$  two-class problems, and computing  $C \times 2$  cost matrices from the  $C \times C$  cost matrix. The procedure for this is described next.

Since the classification system has been trained, a confusion matrix (on training or validation data) can be computed as it would ordinarily be computed in order to evaluate the performance of the classification system. Let the computed  $C \times C$  confusion matrix be

$$\begin{bmatrix} n_{11} & n_{12} & \dots & n_{1C} \\ n_{21} & n_{22} & \dots & n_{2C} \\ \dots & & & \\ n_{C1} & n_{C2} & \dots & n_{CC} \end{bmatrix}$$

where  $n_{ij}$  is the number of instances that belong to Class  $i$  but were classified as Class  $j$ .  $i, j \in \{1, 2, \dots, C\}$ .

Using this confusion matrix, the probability of a misclassified instance classified as Class  $c$  actually belonging to Class  $i$  (one of the type of errors contributing to the false positive rate for Class  $c$ ),  $p_{ic}$  can be computed as

$$p_{ic} = \frac{n_{ic}}{\sum_{j=1}^C n_{jc}, j \neq c} \quad (1)$$

Let the provided  $C \times C$  cost matrix be

$$\begin{bmatrix} 0 & s_{12} & \dots & s_{1C} \\ s_{21} & 0 & \dots & s_{2C} \\ \dots & \dots & \dots & \dots \\ s_{C1} & s_{C2} & \dots & 0 \end{bmatrix}$$

where  $s_{ij}$  is the cost of classifying an instance that belongs to Class  $i$  as Class  $j$ . Here, without loss of generality, the cost of correct classifications,  $s_{ii}$ , is assumed to be zero for all classes.  $i, j \in \{1, 2, \dots, C\}$ .

The expected cost of a false positive for Class  $c$ ,  $E[\phi_c]$  can be computed using the above cost matrix and equation 2

$$E[\phi_c] = \sum_{i=1}^C p_{ic} \cdot s_{ic}, i \neq c \quad (2)$$

Similarly the expected cost of a false negative for Class  $c$ ,  $E[\nu_c]$  can be computed.

Hence, we now have a  $2 \times 2$  cost matrix for Class  $c$  of the form

$$\begin{bmatrix} 0 & E[\nu_c] \\ E[\phi_c] & 0 \end{bmatrix}$$

Doing this for all  $C$  classes, we can break the  $C \times C$  cost matrix into  $C$   $2 \times 2$  cost matrices.

Having done this, the  $C$  optimum thresholds for all  $C$  outputs of the classification system can be determined using the strategies discussed in Subsections 2.1 and 2.2 based on the Bayes classification rule for costs [4] depicted in Fig.1(b).

### 3. EXPERIMENTS AND RESULTS

Experiments were performed to show the statistically significant reduction in the cost by applying dCMS, the applicability of dCMS for a spectrum of datasets of varying complexities and dimensionality, and the applicability of dCMS for classifiers and classification systems of different natures and varying complexities. In order to do so, experiments were conducted in three scenarios.

1. Synthetic data: dCMS was applied to a single classifier, a multi-layer perceptron (MLP) neural network, trained on a simple synthetically generated small dataset.
2. Intrusion detection: dCMS was applied to Learn++ [6], a complex ensembles of classifiers based classification system that performs data fusion and was trained on a large real-world intrusion detection dataset. dCMS was also applied to other standard classifiers such as a Mahalanobis distance based classifier, K-nearest neighbor along with Learn++ [6], however on a dimensionality reduced version of this dataset.
3. Other applications: dCMS was applied to Learn++ [6] on benchmark datasets from the UCI Machine Learning Repository [8] from a variety of other applications such

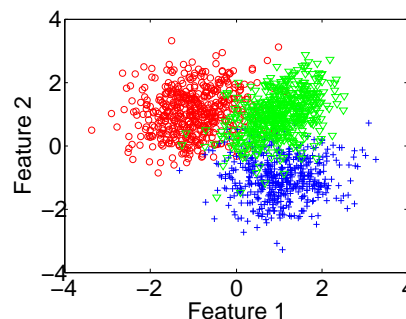


Fig. 2. Distribution of synthetic data points used

as volatile organic compound recognition and optical character recognition.

For all experiments, the following two classification strategies were tested.

1. Without dCMS: The class corresponding to the maximum output of the classification system was selected as the classification decision.
2. With dCMS: The class corresponding to the output that has the maximum (signed) difference from its respective optimum threshold as determined by dCMS was selected as the classification decision.

#### 3.1. Synthetic data

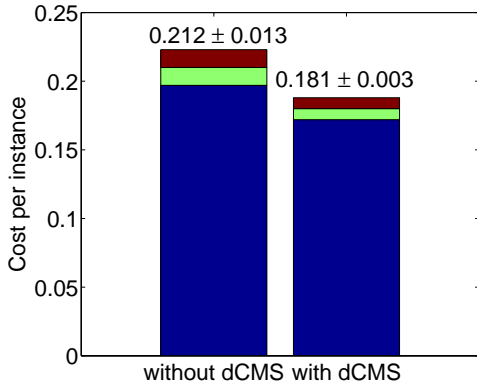
A synthetic dataset of 500 data points were generated from three 2D gaussian distributions with the following parameters, means  $\mu$  and covariance matrices  $\sigma$ . The distribution of the data was as shown in Fig.2.

$$\begin{aligned} \mu_1 &= \begin{bmatrix} -1 & 1 \end{bmatrix}, \sigma_1 = \begin{bmatrix} 0.5 & 0.1 \\ 0.1 & 0.5 \end{bmatrix}, \\ \mu_2 &= \begin{bmatrix} 1 & -1 \end{bmatrix}, \sigma_2 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}, \\ \mu_3 &= \begin{bmatrix} 1 & 1 \end{bmatrix}, \sigma_3 = \begin{bmatrix} 0.4 & 0.2 \\ 0.2 & 0.5 \end{bmatrix} \end{aligned}$$

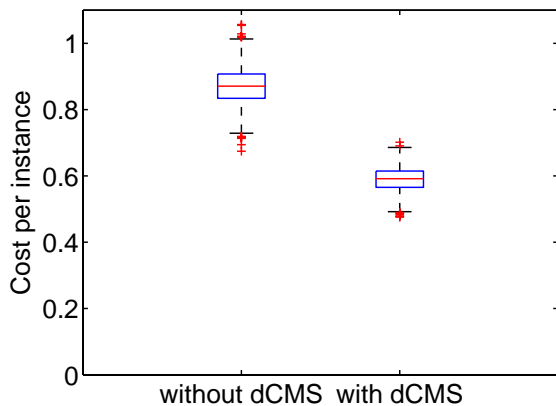
One-third of the points from each class were randomly picked for training, one-third as validation to determine the optimum threshold using dCMS and the rest as testing. An MLP neural network with three output nodes was trained. The outputs were normalized to lie between 0 and 1. dCMS was applied to these outputs to determine the optimum threshold for each output.

The cost matrix assumed was

$$\begin{bmatrix} & class1 & class2 & class3 \\ class1 & 0 & 2 & 4 \\ class2 & 2 & 0 & 3 \\ class3 & 1 & 3 & 0 \end{bmatrix}$$



**Fig. 3.** Results obtained on synthetic data. The cost incurred drops significantly by applying dCMS



**Fig. 4.** Results obtained on an intrusion detection dataset. The cost incurred drops significantly by applying dCMS

This experiment was repeated 100 times (data generated randomly from the same distribution). The results obtained are summarized in Fig.3. It can be seen that the cost incurred after applying dCMS is lower by 17% than that without dCMS with statistical significance. This shows that even on a very easy to classify dataset where an MLP achieves close to Bayesian accuracy, there is room for improvement as far as cost minimization goes because that was not explicitly minimized by the underlying classifier, while dCMS provides that added boost.

### 3.2. Intrusion detection

The MIT-DARPA intrusion detection database [7] is a 41-feature database with over 5 million data points. The 41 features can be grouped into three groups: intrinsic (9), traffic (13) and content (19) features. There are five classes: four for different types of networks attacks - DenialOfService (DOS),

Probe, UserToRoot (U2R), RootToLocal (R2L), and one normal traffic. For each experiment, about 0.3 million were randomly picked for training and validation, and the rest for testing. An ensemble of classifiers based data fusion algorithm, Learn++ [6], inspired in part by Adaboost, which employs several different classifier combination rules to produce the final score, was used. The base classifier used was an MLP neural network. One such classification system was trained for each class involved. Thus the overall classification system has five outputs. dCMS was applied to these five outputs (between 0 and 1). The cost matrix used, as provided with the data was

$$\begin{bmatrix} & DOS & Probe & U2R & R2L & Normal \\ DOS & 0 & 1 & 2 & 2 & 2 \\ Probe & 2 & 0 & 2 & 2 & 1 \\ U2R & 2 & 2 & 0 & 2 & 3 \\ R2L & 2 & 2 & 2 & 0 & 4 \\ Normal & 2 & 1 & 2 & 2 & 0 \end{bmatrix}$$

For statistically significant results, 1000 experiments were conducted with random splits for training, validation and testing. The results obtained are summarized in Fig.4. The cost incurred drops significantly (by 32%) by employing dCMS.

Next, we use PCA to reduce the dimensionality of this data set to 5 feature per feature set (total of 15 dimensions). We used three standard classification techniques for this data. The first was to use the Mahalanobis distance to perform classification, where the classification score of a test instance for each class is inversely proportional to the Mahalanobis distance of that instance from the training data instances from that class. Second, we used a  $K$  nearest-neighbor (KNN) classifier, where the classification score of a test instance for each class is the proportion of its  $K$  training instances neighbors that belong to that class. Third, we used Learn++ [6] which inherently provides a score for each class. dCMS was applied to all these classifiers over random splits of the data into training, validation and testing as described earlier. The results obtained are shown in Table 1. The reductions in cost are by 24%, 19% and 28% for the Mahalanobis distance based, KNN and Learn++ classifiers respectively. This shows the applicability of dCMS to a variety of classifier types, since it is classifier-type independent and only relies on the statistical properties of the classification scores provided by the classifier.

### 3.3. Other applications

We applied dCMS to two other applications. One was volatile organic compounds (VOC) recognition and the other was optical character recognition (OCR). We applied Learn++ [6] as the classification system in both cases. We obtained datasets for these from the UCI Machine Learning Repository [8]. Detailed description of the application of Learn++ [6] to these datasets is given by Parikh and Polikar in [9].

**Table 1.** Results on intrusion detection

Cost per instance	without dCMS	with dCMS
Mahalanobis	$1.83 \pm 0.02$	$1.48 \pm 0.01$
KNN	$1.32 \pm 0.01$	$1.11 \pm 0.01$
Learn++ [6]	$1.24 \pm 0.01$	$0.97 \pm 0.01$

**Table 2.** Results on other applications

Cost per instance	without dCMS	with dCMS
VOC	$0.102 \pm 0.002$	$0.089 \pm 0.001$
OCR	$0.269 \pm 0.003$	$0.201 \pm 0.001$

The VOC recognition task entails recognition of 12 compounds such as Acetone, Hexane, Octane, Toluene, Xylene, Methanol, Ethanol, etc. The dataset has responses from 12 microbalance gas sensors as features. In order to apply Learn++ [6] we randomly split the features into 3 feature sets with 4 features each. The dataset consists of a total of 84 instances, 7 per class, of which we used 2 random instances per class for training, 2 as validation and 3 for testing. Since there was no cost matrix associated with this dataset, we generated a random  $12 \times 12$  cost matrix, with each off diagonal entry uniformly selected from 1 to 5.

The OCR dataset consists of 10 handwritten digit classes. It consists of a total of 649 features, naturally split into 6 different feature sets corresponding to pixel averages, profile correlations, Fourier coefficients, KL coefficients, etc. The dataset consists of 2000 instances, 200 per class of which we used 30 random instances per class as training, 30 as validation and 140 for testing. The  $10 \times 10$  cost matrix was generated in a similar fashion as with the VOC dataset.

The experiments for each dataset were run 100 times with random samplings for the training, validation and testing splits of the data. The results obtained are shown in Table 2. We can see that dCMS reduces the cost with statistical significance for both applications. For VOC, dCMS reduces the cost per instance by 15% while for OCR the reduction is 26%. This shows the applicability of dCMS for a variety of applications and classification problems.

#### 4. CONCLUSION

We propose dCMS, a strategy that gears the classification system towards minimizing the cost incurred due to the classification errors instead of the classification error itself. It can be applied to any classification system that provides a score

for each class. It does not require any re-training of the classification system, and simply exploits the statistical properties of the classifier to make classification decisions more in-tune with these properties and in-light of the costs involved. Statistically significant results have been provided on a variety of applications, data complexities as well as classifier types; ranging from a complex ensemble of classifiers based classification system on a large real-world intrusion detection dataset, to a single MLP neural network trained on a synthetic 2-feature dataset, and several varieties in between. All results demonstrate the effectiveness of dCMS in reducing the classification cost incurred.

Future work involves investigating an iterative form of dCMS. Applying dCMS once to the trained classification system, the classification is performed using the new set of optimum thresholds for each class, and a new confusion matrix is obtained. This can be reused along with the provided cost matrix to determine a new set of optimum thresholds using dCMS again. This suggests an iterative formulation. Convergence issues, and the effectiveness of multiple iterations as compared to the first iteration will be studied.

#### 5. REFERENCES

- [1] A. Jain, A. Ross, S. Prabhakar. An introduction to biometric recognition. In *IEEE Transactions on Circuits and Systems for Video Technology*, 2004.
- [2] N. Lachiche and P. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In *International Conference on Machine Learning (ICML)*, 2003.
- [3] Y. Ma and X. Ding. Robust real-time face detection based on cost-sensitive AdaBoost method. In *IEEE Proceedings of International Conference on Multimedia and Expo (ICME)*, 2003.
- [4] D. Duda, P. Hart and D. Stork. In *Pattern Classification*, 2/e, Chap. 2, pp. 20-29, New York, NY: Wiley Interscience, 2001.
- [5] J. Kittler, M. Hatef, R. Duin and J. Matas. On combining classifiers. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1998.
- [6] M. Lewitt and R. Polikar. An ensemble approach for data fusion with Learn++. In *Proceedings of International Workshop on Multiple Classifier Systems (MCS)*, 2003.
- [7] The UCI KDD Archive, Information and Computer Science, University of California, Irvine, <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- [8] C. Blake and C. Merz. UCI Repository of Machine Learning Database at Irvine CA, 2005. <http://mllearn.ics.uci.edu/MLRepository.html>
- [9] D. Parikh and R. Polikar. An Ensemble-Based Incremental Learning Approach to Data Fusion. In *IEEE Transactions on Systems, Man and Cybernetics*, 2007.